Speechmatics Diarisation Metrics

Introduction

This document described the **sm_diarisation_metrics** package that can be used to evaluate speaker diarization performance. The package can be used to evaluate performance on a single file, or to produce metrics over a complete data set. A number of metrics our output by the tool, including common metrics such as diarization error rate (DER), and all our covered later in this document.

The tool was written on top of the publicly available **Pyannote-Metrics** system. Please see here for more information:

https://github.com/pyannote/pyannote-metrics

Installation

The **sm_diarisation_metrics** package is supplied as a Python wheel. It can be installed as with other python wheels, via pip. For example:

```
pip3 install sm_diarisation_metrics-1.0.0-py3-none-any.whl
```

Other dependencies will be installed at this time (see *Requirements* for the list of packages that will be installed).

Usage

The package has a command line interface that can be used to run evaluation on diarization output. The basic inputs required are a **hypothesis** file, and a **reference** file. The hypothesis file is typically the V2 json output from the batch transcriber (containing ASR transcription results, with speaker labels). The reference file is the ground truth and should give the true speaker segments (either as full speaker segments, or at a word level). Several formats are supported, as covered later in this document.

Evaluating a single file

The first example of how to run is when evaluating performance on a single file. In this case we can assume we have a single reference file (e.g. `reference.json`), and a single hypothesis file (e.g. `transcript.json-v2`). To run the analysis on these files we run a command such as:

```
python3 -m sm_diarisation_metrics.cookbook reference.json transcript.json-v2
```

This leads to output as shown over the page. The output in this case is returned to stdout in a human readable form, giving the values for each applicable metric.

```
python3 -m sm_diarisation_metrics.cookbook reference.json transcript.json-v2

--------------------------------
DER:                    0.227
Insertion:              0.031
Deletion:               0.073
Confusion:              0.123
--------------------------------
Diarization Coverage:   0.987
Diarization Purity:     0.867
--------------------------------
Jaccard Error Rate:     0.735
--------------------------------
Segmentation Coverage:  0.983
Segmentation Purity:    0.928
Segmentation Precision: 0.762
Segmentation Recall:    0.571
Segmentation F1 Score:  0.653
--------------------------------
Word level DER:         0.152
--------------------------------
```

Note that the metrics are given as rates between 0.0 and 1.0. The DER given above for example is 0.227. If considering as a percentage, this is equivalent to 22.7% DER.

Evaluating across a set of files

It is also possible to generate statistics on a set of files, in one command. This has the advantage of producing a csv formatted output file with all the file-by-file results, as well as an overall result containing the average scores across the set. In this case, the averages are computed normalizing by the duration of each file (or number of words in the file, for the word-based metrics). This ensures that the impact any individual file's result has on the overall average result weighted based on the length of the file compared to the total length of audio in the set.

To run across a set of files, to DBL (database list) files need to be created. These are simple text files listing the full paths to each of the reference files (for the reference DBL) or hypothesis files (for the hypothesis DBL). Note that the order of the files in the two lists must match – and that there are the same number of files in each. For example:

```
/work/data/reference/conference1.json
/work/data/reference/interview1.json
/work/data/reference/media1.json
```

```
/work/results/conference1/transcript.json-v2
/work/results/interview1/transcript.json-v2
/work/results/media1/transcript.json-v2
```

With the DBL list files created, the command to evaluate can now be run:

```
python3 -m sm_diarisation_metrics.cookbook reference.dbl hypothesis.dbl --output-format
csv --outdir results
```

This is specifying the DBL files to use (the list of reference files, and the list of hypothesis files), the output format (`csv` in this case, can also be json) and the output directory. Running this command will simply result in the progress of the evaluation being displayed – no actual scores or other metrics will be given to *stdout*. However, the results will be written out in detail into either a CSV or JSON file within the specified output directory (which must exist prior to running the command).

In the above example, the output format has been set to `csv`. In this case, we can find a file `results.csv` in the output directory. This can be loaded into Excel (etc), or otherwise processed. An example of the contents is shown below:

```
Ref_Lab, Hyp_Lab, Duration, DER, Insertion, Deletion, Confusion, Diarisation_Purity,
Diarisation_Coverage, Jaccard_Error_Rate, Segmentation_Purity, Segmentation_Coverage,
Segmentation_Precision, Segmentation_Recall, Segmentation_F1_Score, Word_Level_DER
/data/reference0.lab, /results/test0.lab, 80, 0.77, 0.25, 0.00, 0.52, 0.61, 0.66, 0.74,
0.65, 0.88, 0.25, 0.08, 0.12, 0.20
/data/reference1.lab, /results/test1.lab, 80, 0.40, 0.02, 0.01, 0.38, 0.85, 0.62, 0.50,
0.96, 0.98, 1.00, 0.85, 0.92, 0.54
/data/reference2.lab, /results/test2.lab, 80, 0.44, 0.18, 0.14, 0.13, 0.86, 0.86, 0.44,
0.81, 0.81, 0.55, 0.46, 0.50, 0.00
/data/reference3.lab, /results/test3.lab, 80, 0.11, 0.01, 0.02, 0.08, 0.92, 0.92, 0.20,
0.98, 0.97, 1.00, 1.00, 1.00, 0.07
Overall_Result, , 320, 0.43, 0.11, 0.04, 0.27, 0.81, 0.76, 0.47, 0.85, 0.91, 0.70, 0.60,
0.63, 0.20
```

Each line represents the results for a specific file, except for the first line (which are the column titles) and the last line (which is the set average scores). If the output format is set instead to **json** then the output file will be called `results.json`.

Supported Formats

The input reference and hypothesis files may be specified in a number of different format. The tool currently supports lab, ctm and json formats. In the latter case, there are two different definitions that may be used.

LAB format

This is a simple format that just gives a sequence of segments, each of which has a start time, the end time, and the speaker label. An example of the **lab** format is:

```
0.000 0.500 speaker_1
0.650 2.500 speaker_2
2.930 4.820 speaker_1
```

In the above the fields are `<start_time>`, `<end_time>` and `<label>`. Input label files must have the ".lab" file extension.

JSON format

There are two json file definitions that may be used as input for evaluation. The first is the Speechmatics V2 Transcription format, such as produced by the transcriber. As such, this is perhaps the most convenient input format for the hypothesis, as this is how it is output by the system. The specification of this file format is documented elsewhere, and no further details are given here.

The other format supported, more commonly used as a reference file, is given as follows:

```
[
    {
        "speaker_name": "Speaker 1",
        "start": 0.0,
        "duration": 0.500
    },
    {
        "speaker_name": "Speaker 2",
        "start": 0.650,
        "duration": 1.850
    }
]
```

Extra fields (such as words, etc) are valid within the json file, but only the fields shown above will be used.

CTM format

This is another simple format that is not dissimilar to the **lab** format given previously. In this case though, there are a few extra fields (which are not actually used for the diarization evaluation, but must be specified to preserve the format). Also, the duration is given instead of the end time for each segment. An example:

```
1 A 0.000 0.500 speaker_1 0.000
1 A 0.650 1.850 speaker_2 0.000
1 A 2.930 1.890 speaker_1 0.000
```

In this case the fields are:

- *Channel (unused)*
- *Segment (unused)*
- Start time
- End time
- Speaker label
- *Confidence (unused)*

Input CTM files must have the ".ctm" file extension.

Metrics

In this section the metrics produced are described in more detail.

Diarization Error Rate

This is one of the key metrics used in evaluating a diarization system. It is computed based on three other key error rates:

- Diarization insertion rate
- Diarization deletion rate
- Diarization confusion rate

The formula for computing the DER is:

$$DER = \frac{\text{confusion} + \text{miss} + \text{false alarm}}{\text{total reference speech time}}$$

It can be shown in a diagram such as:

Note that all of these metrics are "duration" based – that is they are computed based on the duration of the file that is in error compared to the total duration of speakers in the reference. It is quite sensitive to mismatches in speech/non-speech between the hypothesis and the reference file. Also, these metrics aren't so powerful in identifying errors that may be small in terms of the percentage of audio that's effected, but can be serious in terms of segmentation (e.g. entirely missing a speaker that only appears infrequently in a file).

Jaccard Error Rate

The Jaccard Error Rate (JER) is similar to the DER as described above. However, in this case the error rate is compute for each speaker, and the overall error rate from a file is then based on the average error rate per speaker. This metric does allow for errors in insignificant speakers (those that only occur for a relatively short period within the file) to be made more visible, as all speakers are given the same weight.

Word Diarization Error Rate

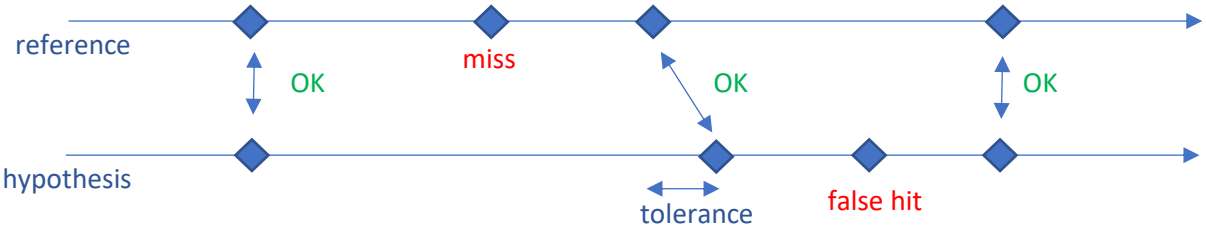
This metric shows the error rate at a word level. For all words in the hypothesis transcript, it is determined whether or not that word has been labelled correctly (based on which speaker label the word overlaps with most in the reference). The word diarization error rate (WDER) is then computed as:

$$WDER = \frac{\text{incorrect words}}{\text{total hypothesized words}}$$

Segmentation Metrics

The segmentation metrics capture the quality of the segmentation, as in the correct positioning of the speaker change points, etc. It does not make use of the diarization labels, only the points that speakers are changing. Primary amongst the segmentation metrics are the **segmentation recall** (the rate at which change points given in the reference are correctly detected in the hypothesis), **segmentation precision** (the proportion of change points determined in the hypothesis that are correct), and the **F1 Score** metric (which is a balance of the precision and recall).

Understanding the meaning of these metrics can be helped via the following diagram:



When matching the change points between the reference and the hypothesis, a certain degree of *tolerance* is allowed (i.e. difference in the time position of the matching points). By default this is set to 1.0 seconds, but can be configured using the `--segmentation-tolerance` flag.

The definition of segmentation recall is:

$$Recall = \frac{\text{correct hypothesized changes}}{\text{total reference changes}}$$

The definition of segmentation precision is:

$$Precision = \frac{\text{correct hypothesized changes}}{\text{total hypothesized changes}}$$

The definition of the segmentation F_1 Score is:

$$F_1\ Score = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

Diarization Purity and Coverage

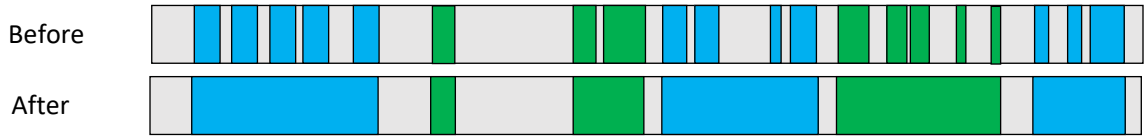
These are further metrics that are provided on the diarization performance. These have been exposed from Pyannote, but not used as a primary metric for optimization. More information on these metrics can be found in the Pyannote documentation.

Notes on input pre-processing

As already specified, the input hypothesis and reference segments are read in from several different file formats. There are a couple of processes that may be run on these inputs, depending on the file type, prior to the metrics being computed. These are all performed automatically within the tool – the user doesn't have to worry about this. However, the processes are documented here for informational purposes.

Inter-Segment Gap Removal

Gaps up to a certain duration between segments of the same speaker are removed, combining these segments to form a single segment. This is to reduce mismatch between the hypothesis and reference in terms of how small non-speech segments are handled. Currently the maximum gap that will be removed is 5.0s (this is not currently configurable).



The most common reason that it is required to do this is that whilst the input may contain segments that are at speaker turn level (so long segments, that could cover many words), it may alternatively contain lots of smaller segments, at the word level. This is typically the case for the hypothesis when taken from a Speechmatics V2 json transcription file. With the mismatch in how gaps are represented (or not), then the "duration" based metrics will suffer many insertion and deletion errors that are not actually a genuine reflection on the system. We also need to merge in this way when looking at segmentation metrics (change point detection, etc), as having gaps within a single speaker "block" can register as false change points.

Currently whether merging is performed or not depends on both the input file type, and the metric being computed:

	LAB / CTM Reference	**JSON Reference**	**LAB / CTM Hypothesis**	**JSON Hypothesis**
Diarization Metrics	No	Yes	No	Yes
Segmentation Metrics	Yes	Yes	Yes	Yes
Word Level Metrics	Yes	Yes	No	No

Note, the reason that json files are treated differently is as these are typically expected to be at the word level, and thus require merging to create speaker segments.

Removal of Unknown Speaker

Speechmatics V2 transcription output uses a special unknown speaker label ("UU") on words for which it was unable to determine the speaker. These are only expected in **json** input into the system, and as such the removal of such terms is only carried out when the input type is **json**. These are preserved whilst merging segments (to prevent two segments of the same speaker that are split by an "unknown" speaker segment from being merged) but are removed prior to computation. This is because these labels should not be treated as truly identified speakers and should be classed as a **diarization deletion** error assuming there is a speaker in the reference.

Advanced Options

Aside from passing in the reference and hypothesis files, there are several other options that can be configured. The following table shows the list of extra parameters / flags that may be used:

Parameter	Value Type	Description
--output-format	String	The output format in which to write the results. Used when evaluating a full data set (having passed in DBL files listing the reference and hypothesis files). Permitted values: **csv** or **json**
--segmentation-tolerance	Float	The tolerance (in seconds) when matching speaker change points for segmentation metrics. This is how far off the hypothesized point can be from the reference point and still be counted as a true hit. Default is **1.0**.
--outdir	String	The output directory into which to write the results files (when evaluating over a set).
--show-words	Boolean	If this is set, then the output word-by-word results will be displayed. At present this option is only available when running evaluation on a single file. The output will be of the following format: `---------- Word level diarization information: [WDER] 0.000 0.500 OK [WDER] 0.600 1.100 OK [WDER] 1.300 1.900 OK [WDER] 2.300 2.800 OK [WDER] 2.800 3.100 OK [WDER] 3.100 4.000 OK [WDER] 4.200 5.000 ERROR`

Requirements

The python wheel depends on the following packages, which should be picked up during installation:

- Flask (version: 1.1.1)
- requests (version: 2.22.0)
- structlog (version: 19.2.0)
- pyannote.core (version: 3.7.1)
- pyannote.database (version: 3.0.1)
- docopt (version: 0.6.2)
- tabulate (version: 0.8.7)
- scikit-learn (version: 0.22.2.post1)